Data Manipulation Language (DML)
(لغة معالجة البيانات)

SELECT .... (إختيار)
INSERT .... (أضافه)
UPDATE .... (تحديث)
DELETE .... (حذف)

Data Definition Language (DDL)
(لغة تعريف البيانات)

CREATE TABLE .... (إنشاء جدول)
DROP TABLE .... (حذف جدول)
GRANT .... (صلاحيات)

* Syntax *

SELECT < قائمة الأعمـــــــــــدة >
FROM < قائمة الجداول >

TABLES1

P#	Name	Qty
1	Bolt	25
2	Nut	42
3	Wheel	15

EX1:في هذا المثال نحدد الأعمدة المطلوبة

SELECT P#, Qty
FROM TABLES1

P#	Qty
1	25
2	42
3	15

EX2:في هذا المثال سوف يظهر لنا جميع الأعمدة

SELECT *
FROM TABLES1

P#	Name	Qty
1	Bolt	25
2	Nut	42
3	Wheel	15

*** الاختيار لكن بشرط ***

SELECT < قائمة الأعمدة >
FROM < قائمة الجداول >
WHERE < قائمة الشروط >

الرموز المستخدمة في الشرط هي

[= , < , > , < > , <= , >=]

SELECT *
FROM TABLES1
WHERE P# = 2

P#	Name	Qty
2	Nut	42

العمليات الجبرية

AND , OR , NOT

EX3: في هذا المثال أم أن يتحقق الشرط الأول أو الثاني أو كلاهما

SELECT *
FROM TABLES1
WHERE P# = 1 OR P# = 2

P#	Name	Qty
1	Bolt	25
2	Nut	15

EX4:في هذا المثال يجب أن يتحقق الشرطين

```
SELECT *
FROM TABLES1
WHERE Name = 'Wheel' AND Qty > 10
```

P#	Name	Qty
3	Wheel	15

يمكنك استبدال هذه الصيغة بطريقه أخرى باستخدام IN

[WHERE (P# =1)OR(P# =2)OR(P# =3)]

```
SELECT Name
FROM TABLES1
WHERE P# IN (1 , 2 , 5)
```

Name
Bolt
Nut

لتسهيل الموضوع عند اختيار أعدد معينه في نطاق معين نستخدم

[BETWEEN]

```
SELECT *
FROM TABLES1
WHERE Qty BETWEEN 20 AND 50
```

متطلبات شائعة لحساب إحصائيات مثل

[MIN (أصغر) , MAX (أكبر) , AVG (متوسط)]

[SUM (الجمع) , COUNT (العداد)]

```
SELECT MIN(Qty) , MAX(Qty) , AVG(Qty)
FROM TABLES1
WHERE Name = 'Wheel'
```

………………………………………………………….

الخواص الحسابية

ملاحظه: العمليات الحسابية توضع في SELECT فقط

(+ , - , * , /)

```
SELECT Name , Qty + ( Qty * 0.2 )
FROM TABLES1
```

Name	Qty
Bolt	30
Nut	50.4
Wheel	18

الترتيب

[ORDER BY < قائمة الخواص >]

ملاحظه : لجعل الترتيب تنازلي نضع DESC بعد قائمة الخواص

```
SELECT Name , Qty
FROM TABLES1
ORDER BY Qty
```

Name	Qty
Wheel	15
Bolt	25
Nut	42

....

لاستعراض كل المعلومات للأسماء التي تبدأ بحرف نختاره

```
SELECT *
FROM TABLES1
WHERE Name LIKE ' D% '
```

في هذا المثال سوف يستعرض جميع المعلومات للأسماء التي تبدأ بحرف D

الحرف الأول W والحرف الأخير L

WHERE Name LIKE ' W%L '

في المثال التالي الاسم مكون من ٣ حروف الحرف الأول هو Z

WHERE Name LIKE ' Z - - '

توضع علامة % عندما يكون الاسم طويل فقط

* مراجعات *

```
SELECT SUM(Qty + Qty * 0.25)
FROM TABLES1
```

وضعنا العملية الحسابية في الشرط لأننا نشترط أن تساوي العملية الحسابية القيمة ٥٠

```
SELECT *
FROM TABLES1
WHERE Qty + Qty * 0.25 = 50
```

المجموعات
[GROUP BY]

TABLES2

EMP	Name	Salary	FUN
1	Fred	200	Sales
2	Mike	300	Sales
3	Sam	400	Sales
4	Martha	350	Marketing
5	Juanita	500	Marketing
6	Steve	800	Finance
7	Tom	200	Service
8	Sue	900	Service

```
SELECT  FUN , AVG(Salary)
FROM TABLES2
GROUP BY FUN
```

FUN	AVG(Salary)
Sales	300
Marketing	425
Finance	800
Service	550

في هذا المثال يستعرض جميع الأسماء ماعدا التي تبدأ بحرف S

```
SELECT  FUN , MAX(Salary)
FROM TABLES2
WHERE Name NOT LIKE ' S%'
GROUP BY FUN
```

FUN	MAX(Salary)
Sales	300
Marketing	500
Service	200

كيف نستخدم العمليات الأخصائية في الشرط باستخدام

HAVING < قائمة العمليات >

```
SELECT FUN , AVG(Salary)
FROM TABLES2
GROUP BY FUN
HAVING COUNT(*) > 3
```

العمليات الأحصائية مثل

[SUM ,MIN , MAX , AVG , COUNT]

توضع في HAVING إذا استخدمناه كعمليه شرطيه غير ذالك

توضع في SELECT

EXP: مثال أخرى

```
SELECT FUN , MIN(Salary)
FROM TABLES2
WHERE FUN IN (1,2,3,4,5)
GROUP BY FUN
HAVING AVG(Salary) > 100
```

العـــــــــــلاقات

PRIMARY KEY ….. P.K

شروطها

لا يكون فارغ (NULL)

ولا يأخذ قيم متكررة

FOREIGN KEY ….. F.K

TAB3

EMP#	Name	Salary
1	Fred	200
2	Ethel	300
3	Mike	400

P.K

TAB4

EMP2#	Name2
1	Harvard
2 2	IIT Michigan
3 3 3	MIT Stanford IIT

F.K

EXP : في هذا المثال سوف نوضح كيف نربط بين جدولين

```
SELECT  TAB4.Name2
FROM  TAB3 , TAB4
WHERE  TAB3.EMP# = TAB4.EMP2#
     AND  TAB3.Name = 'Mike'
```

لاحظ عندما نريد معلومة من عامود نكتب أسم الجدول . ثم أسم العامود

في المثال السابق لاحظ أننا أشترطنا أنا القيمة في العامود الذي
أسمه(EMP#) في الجدول الثالث
تساوي نفس القيمة في العامود الذي
أسمه(EMP#) في الجدول الرابع
+ العامود الذي أسمه Name في الجدول الثالث يساوي قيمه

يمكن أن تربط أكثر من جدول مع بعضها
وهذه الطريقة هي التي تستخدم في ربط الجداول عند أنشأها في البرنامج

```
PRIMARY KEY(EMP#)
FOREIGN KEY(EMP2#) References
TAB3(EMP#)
```

....

Subqueries

هي بعبارة واضحة SELECT داخل SELECT

EXP :

```
SELECT *
FROM TAB3
WHERE EMP# IN ( SELECT EMP2#
                FROM TAB4 )
```

Exists

عندما يتم تنفيذ SELECT الداخلي ويظهر لنا نتائج تكون
(TRUE) فيتم تنفيذ SELECT الأخرى
أما إذا كانت النتيجة (FOULS) فلا ينفذ شي

```
SELECT *
FROM TAB3
WHERE  EXISTS ( SELECT TAB4.EMP2#
                FROM TAB4
        WHERE TAB4.EMP2# = TAB3.EMP# )
```

....

Distinct

في هذا الموضع جميع الأشياء المكررة لا يطبعها بمعنى لو كان هناك عدد مكرر أكثر من مره فسوف يظهره مره واحده فقط

EXP :

```
SELECT DISTINCT ( Salary )
FROM TAB3
```

UNION

وهي عملية الدمج أو الجمع للقيم المخرجة

EXP :

```
    SELECT *
    FROM TAB3
UNION
    SELECT *
    FROM  TAB4
```

....

OTHER DATA MANIPULATION

معالجة بيانات أخرى

UPDATE .... تحديث

DELETE .... حذف

INSERT .... إضافة

Syntax:
UPDATE < أسم الجدول>
SET <أسم العامود> = < القيمة >

EXP :
UPDATE TAB3
SET Salary = 150.00
WHERE Name = 'Fred'

EXP :
UPDATE TAB3
SET Salary = Salary * 1.10

Syntax :
DELETE FROM < أسم الجدول >
WHERE < الشرط >

EXP :في هذا المثال يتم حذف كل أسم يأخذ مايك
DELETE FROM TAB3
WHERE Name = 'Mike'

EXP :في هذا المثال سوف يحذف الجدول كامل
DELETE FROM TAB4

Syntax :

INSERT INTO < أسم الجدول > (< أسماء الأعمدة >)
VALUES (< قيم الأعمدة >)

EXP :
INSERT INTO TAB5 (NUM , Name , Salary)
VALUES (1, 'MAJED',5000)

-

Data Definition Language (DDL)
(لغة تعريف البيانات)

CREATE TABLE < أسم الجدول >
(< أسم العامود > < نوع البيانات > ,
< أسم العامود > < نوع البيانات >)

أنواع البيانات
Char (x) , VARCHAR(x) , SMALLINT ,
INTEGER , DATE , TIME , DECIMAL(x,y)

Exp : هذا المثال يوضح كيف يمكن أنشاء جدول

```
CREATE TABLE  TAB6
( EMP#          SMALLINT,
  NAME          CHAR(20),
  SALARY        DECIMAL(5,2))
```

EXP : لا يمكن تركه فارغاً

CHAR (30) NOTNULL

الفرق بين CHAR (x) & VARCHAR(x)

CHAR (١٠) :-
لو تم حجز ١٠ حروف فإنها سوف تحجز مكان حتى ولو لم تستخدم هذه الخانة

GOOD_ _ _ _ _ _ _

VARCHAR (١٠) :-
لو تم حجز ١٠ حروف فإنها سوف تحجز مكان على حسب القيمة المدخلة أو المستخدمة

GOOD

EXP : اختيار كل المعلومات في العامود المحدد ما عدى الخالية أو بمعنى أخرى التي لا توجد لها قيمة

```
SELECT *
FROM TAB6
WHERE Salary IS NULL
```

DROP TABLE

DROP TABLE < أسم الجدول المراد حذفه >

ALTER TABLE

Syntax:

إضافة عامود جديد لجدول سابق

ALTER TABLE < أسم الجدول >
ADD (< أسم العامود > < نوع البيانات > ,
< أسم العامود > < نوع البيانات >)

تعديل عامود

ALTER TABLE < أسم الجدول >
MODIFY (< أسم العمود > < نوع البيانات >)

تغيير أسم الجدول

ALTER TABLE < أسم الجدول القديم >
RENAME TO < أسم الجدول الجديد >

SECURITY AND AUTHORIZATION
(الأمن و التفويض)

Syntax:

GRANT [ALL , SELECT , INSERT , UPDATE , DELETE]
ON < أسم الجدول > TO < أسم الشخص المخول له >

EXP :

GRANT ALL ON TAB6 TO Majed

EXP :يمكن أعطى الصلاحية لأكثر من شخص

GRANT SELECT , DELETE ON TAB5 TO Majed , Ahmad

EXP :يمكن أعطى الصلاحية لعامود فقط

GRANT UPDATE(Salary) ON TAB6 TO Majed

EXP :يمكن أعطى الصلاحية لكل المستخدمين

GRANT ALL ON TAB4 TO PUBLIC

REVOKE

(إلغاء الصلاحيات)

REVOKE < أسم التعليمة > ON < أسم الجدول > FROM < أسم الشخص >

REVOKE DELETE ON TAB4 FROM Majed

--*-*-*-*-*-*-*-*-*-*-*-*-*-*-**-*-*-*

أعداد: KSA

www.phpvillage.com